AFIT/GOR/ENS/94M-14

An Efficient Approach to Solving the Optimal Control of Arrivals Problem

THESIS

John Raymond Simeoni
Captain, USAF

AFIT/GOR/ENS/94M-14

AFIT/GOR/ENS/94M-14

An Efficient Approach to Solving the Optimal Control of Arrivals Problem

THESIS

Presented to the Faculty of the Graduate School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Operations Research

John Raymond Simeoni, B.M., M.A., M.B.A.

Captain, USAF

March, 1994

THESIS APPROVAL

STUDENT: John R. Simeoni, Captain, USAF CLASS: GOR-94M

THESIS TITLE: An Efficient Approach to Solving the Optimal Control of Arrivals Problem

DEFENSE DATE: 24 February 1994

COMMITTEE:

Name/Title/Department	Signature
Advisor: DENNIS C. DIETZ, LTC, USAF Assistant Professor of Operations Research Department of Operational Sciences	Dennis C Dietz
Reader: PETER W. HOVEY, PHD Assistant Professor of Statistics Department of Mathematics and Statistics	Peter W. Hovey

Acknowledgements

First, I must thank my wife Cynthia and son Michael for their support and understanding during the months spent writing this thesis. Additionally, I owe a great deal of thanks to my advisor, Lt Col Dietz, for his insight and guidance without which this work would not have been accomplished. I also wish to thank my reader, Dr Hovey for his advice and help in completing this work.

John Raymond Simeoni

Table of Contents

	Page
Acknowledgements	iii
Abstract	vi
I. Introduction	1
II. Literature Review	3
III. Theoretical Results	5
3.1 Overview	5
3.1.1 Assumptions	5
3.1.2 Notation	5
3.1.3 Definitions	6
3.1.4 Problem Description	7
3.2 Late-Start Scheduling	8
3.3 Clustering	9
IV. Foundation	10
V. Algorithm	14
5.1 Methodology	14
5.2 Stopping Criteria	15
5.3 Sample Problem	18
VI. Examples	20
6.1 k=15, N=40	20
6.2 k=20, N=16	21
6.3 k=32, N=25	21

Page

6.4 Server Shuts Down, k=20, N=16 22

6.5 Continuous Case, k=7, N=5, μ=1, C_s=4 23

VII. Conclusions and Recommendations 24

Appendix A. Alternate Proofs 25

A.1 Alternate Proof of Theorem 1 25

A.2 Alternate Proof of Corollary 1 25

Appendix B. Computer Code 27

Bibliography 28

Vita 29

AFIT/GOR/ENS/94M-14

Abstract

The optimal control of arrivals problem is one which has many applications in both defense and industry. Simply stated, the problem addresses how to schedule a finite number of customers in a finite number of equal-length time slots, where each customer's service time comes from a specified probability distribution. There are two cost components, one based on total expected customer waiting time and the other based on the expected amount of time the server stays open beyond its scheduled completion time. Currently, solutions have been developed to the optimal control of arrivals problem but they are computationally slow and only work for exponential distributions. This thesis presents an algorithm for the optimal control of arrivals problem which is both computationally efficient and works for r-Erlang distributions.

An Efficient Approach to Solving the Optimal Control of Arrivals Problem

I. Introduction

This thesis presents a solution to the optimal control of arrivals problem, which encompasses the areas of planning, scheduling, and control of queues. The problem addresses how we schedule a finite number (N) of appointments in a finite time horizon which is divided into a finite number (k) of equal length time units. The objective is to determine the schedule with the minimum expected cost, where cost is measured in two components: waiting time cost and overtime cost. Waiting time cost is the amount of expected customer waiting time multiplied by the waiting time cost per unit. Overtime cost is the expected amount of time that the server works beyond the end of the time horizon multiplied by the overtime cost per unit. We assume that all customers arrive exactly on time as scheduled, and the server works on a first-in, first-out basis. We also assume that customer service time follows an r-Erlang distribution. In all cases, we assume the service time parameter, μ, is defined as the service rate per time unit Δ, where Δ is the length of each of the k time intervals. Wang (1) showed that the actual waiting cost and actual overtime cost need not be considered, for only their ratio is important. That ratio is what we will refer to as C_s, or the cost factor. Therefore in all calculations, we assume waiting time cost is standardized to 1, while overtime cost equals expected overtime multiplied by the cost factor.

The problem has many real-world applications. One such case is in the area of shipping. Consider a shipping company which has leased several hours of dock access to load/unload some of its ships. It costs money to have a ship waiting in the harbor before it can dock. Also, there will usually be a high penalty for exceeding the lease period, because the company that owns the dock wants to ensure the next lessee is able to start on time. Therefore, assuming the shipping company has paid a fixed fee for use of the dock, the only additional costs will be the cost for ships waiting in the harbor, and the cost for usage

time beyond the end of the lease period. A similar problem can arise in the context of airfield operations when cargo handling resources are limited.

Much research has been done on the optimal control of arrivals problem, but the solutions developed are computationally difficult. Furthermore, the solutions have been approached primarily from a numerical standpoint. We will develop an iterative procedure for determining the optimal schedule which will require the evaluation of only a small subset of all feasible schedules. The algorithm is efficient and provides instant sensitivity analysis on the optimal solution.

The algorithm works for r-Erlang service distributions, which are defined by the probability density function:

$$f(t) = \lambda e^{-\lambda t} * (\lambda t)^{r-1}/(r-1)!, t \geq 0$$

Being able to solve problems using r-Erlang distributions allows one to match the first two moments from data, hence the algorithm is both a valuable and practical tool to use for real-world applications.

II. Literature Review

The optimal control of arrivals problem has received much attention since the work of Naor (2) was published in 1969. In his article, he addressed the idea of limiting queue length based on both individual benefit (the benefit of the arriving customer) and social benefit (the benefit of the entire population). He presented solutions to both problems for the $M/M/1$ queue. Knudsen (3) extended Naor's results to the $M/M/k$ queue, and Yechiali extended the results to first the $G/M/1$ queue (4) and later the $G/M/k$ queue (5). Rue and Rosenshine developed optimal control policies for both the $M/M/1$ (6) and $M/E_k/1$ (7) queues serving multiple classes of customers, where each class has its own cost-reward structure.

Pegden and Rosenshine originally posed the optimal control of arrivals problem. The original problem considered the planning of N arrivals over an infinite continuous time horizon, where cost consisted of two components: customer waiting time and server completion time. The service time was assumed to be exponential. An analytic closed form solution was presented for the specific case of 2 arrivals. In this case, the objective function was shown to be convex. However, no closed-form solution could be found for the case where $N > 2$, and moreover, the objective function for the general case has not yet been proven to be convex. Numerical methods were employed to determine optimal arrival times for the case of $N > 2$, but these optima were only guaranteed to be local optima due to the uncertainty of the convexity of the objective function. Healy, Pegden, and Rosenshine (8) extended the results of the single, exponential service time system to two parallel servers with exponential service time servers.

Because a closed-form solution to the original problem seemed intractable, the problem was redefined with a finite, discrete time-unit time horizon, divided up into k equal length time slots. Instead of scheduling an appointment at any point in the time window, it would now have to be scheduled at the beginning of one of the k time slots. Server cost was measured as overtime, which is the expected amount of time that the server works beyond the end of the finite time window times the cost per unit of overtime. As before, it was assumed that all customers arrive exactly at the time of their scheduled appointment. It is important to note that this revised version of the problem is actually more reflective

of the real world than the original, due to the use of the finite time window and finite number of possible scheduled appointment times.

Two versions of the problem were posed. First, a myopic, or short-range version where the schedule is evaluated at the beginning of each of the k time slots. This is a dynamic approach, where suboptimizations are performed k times. The other problem is the long-term approach. In this case, the entire schedule is set up before the first arrival, and cannot be revised at any later point in time. This version better reflects real-world scheduling, since one normally schedules a block of appointments at one time (e.g. one day's worth) and seeks to optimize the expected cost for that period of time.

Liao (9) presented solutions to both the short-range and long-range versions of the problem with exponentially distributed service time. The short range problem was solved using dynamic programming, and then the long range problem was solved using Branch and Bound with the short range optimal solution taken as the initial lower bound. In the same work, Liao extended the results to the following models:

- One server with Erlang distributed service time.
- Multiple parallel servers with exponentially distributed service time.
- Two servers in sequence, each with exponentially distributed service time.
- One server with exponentially distributed service time where the server shuts down after the last appointment has completed service.
- Several classes of customers in the system, each class having different waiting time distributions and waiting cost structures.

Wang (1) has recently solved both the short range and long-range versions of the original continuous problem using phase-time distributions. The number of appointments (N) was given, but an infinite continuous time horizon was used. Cost was measured in both waiting time and server completion time. Service time for his solution was assumed to be exponential.

III. Theoretical Results

3.1 Overview

Upon reviewing the previous research on the optimal control of arrivals problem, we see that in general a closed form solution is intractable. Past methodology has employed numeric methods to determine optimal schedules, and certain unproven yet intuitive assumptions have been made in the algorithms. We shall take a more direct approach to the solution of the problem. Before we begin our description of our model, we will introduce the notation and definitions found throughout this work.

3.1.1 Assumptions.

Assumption 1 Each of the k time intervals are of equal length. When service time is exponential, the service rate μ is expressed in customer service completion per time interval.

Assumption 2 The service rate for each customer is μ. When the service distribution is r-Erlang, the service rate for each of the r stages of customer service is $r\mu$.

3.1.2 Notation.

$k =$ the number of time intervals

$N =$ the number of appointments to be scheduled

$\mu =$ the service rate

$\Delta =$ the length of each time slot

$\Theta =$ the length of the finite time horizon

$S, S_s =$ schedules

$T, T_s =$ subschedules

$t_i =$ the i^{th} time slot of a schedule

$a_j =$ the j^{th} customer of a schedule

$b_i =$ the i^{th} component of $S =$ the number of appointments scheduled at t_i

s_j = the scheduled time of the j^{th} customer

C_w = cost per unit of waiting time

C_o = cost per unit of overtime

C_s = the cost factor = C_o/C_w

W$[S]$ = the expected total waiting time of S

W$[T]$ = the expected total waiting time of T

O$[S]$ = the expected overtime cost of S

O$[T]$ = the expected overtime cost of T

$\tau[S]$ = the total expected cost of S

W$[a_i]$ = expected waiting time of the i^{th} appointment

O^* = expected waiting of all appointments succeeding a particular appointment plus O$[S]$

m-cluster = m appointments scheduled in the same time unit, $b_i = m$

3.1.3 Definitions.

Definition 1 A schedule S is a *candidate* if it has not been eliminated from consideration as being the optimal schedule. Initially, all schedules are candidates.

Definition 2 The number of appointments scheduled in the i^{th} time slot of S, t_i, is the i^{th} *component* of S.

Definition 3 T is a *subschedule* of S if T represents any consecutive sequence of components of S.

Definition 4 T is an *origin subschedule* of S if T is a subschedule of S and the first time slot of T represents the first time slot of S.

Definition 5 T is a *terminating subschedule* of S if T is a subschedule of S and the last time slot of T represents the last time slot of S.

Definition 6 T_1 and T_2 are *relative* if they have the same number of time slots and the same number of appointments.

Definition 7 T_1 and T_2 are *perfectly relative* if they are relative, the first appointment of T_1 is scheduled at the same time as the first appointment of T_2, and last appointment of T_1 is scheduled at the same time as the last appointment of T_2.

Definition 8 T_1 *precedes* T_2 if every appointment of T_1 is scheduled at the same time as or before its corresponding appointment of T_2, with at least one appointment of T_1 being scheduled before its corresponding appointment of T_2.

3.1.4 Problem Description. The simplest way to solve the optimal control of arrivals problem would be to enumerate all possible solutions in order to find the best one. However, for all but the smallest problems, complete enumeration is computationally impractical. Given N appointments to schedule in k time units, the number of possible schedules is given by:

$$\binom{N+k-1}{k-1}$$

This is the classic "N balls in k cells" partioning problem (10) which one studies in any introductory probability class. For example, a problem where $N = 5$ and $k = 7$ has 462 possible schedules. A problem where $N = 8$ and $k = 10$, has 24,310 schedules. A larger problem with 25 appointments and 32 time units has over 5.57 x 10^{15} possible schedules, so complete enumeration is not practical for large problems. We will show that we can quickly eliminate groups of schedules which cannot be optimal, and thus will evaluate a much smaller number of schedules prior to determining the optimal solution. We now consider reasons why certain schedules can never be optimal. Throughout this paper, we will refer to *Example A* as the case where $N = 5$, $k = 7$, $\mu = 1$, and $C_s = 4$.

First of all, whether or not a schedule can be optimal is dependent on the value of the service rate, μ, and the cost factor, which is the ratio of overtime cost to waiting time cost. For example, consider Example A. We will represent a typical schedule, such as one with one appointment scheduled in each of the first five time intervals as follows:

[1111100]

Now consider the situation for Example A where μ is arbitrary and customer waiting cost is zero (the cost factor C_s is infinite). In this case, the optimal schedule is the one that minimizes expected overtime, which is obviously:

[5000000]

One may think of this as a model of a doctor's scheduling system. Doctors often schedule all appointments at the beginning of a time period. Thus they are making the tacit assumption that the ratio of the value of their time (or overtime) to the value of the patient's time is infinite.

Conversely, if overtime has no cost associated with it, we would schedule the appointments so as to minimize expected waiting time. Intuitively we would expect the optimal schedule to be one in which the appointments are fairly evenly spread out throught the time window (although the actual optimal schedule will depend on the value of μ). Imagine a company clinic which is staffed by salaried nurses. Each worker that waits at the clinic costs the company money since he/she is not working, and is not receiving treatment. However, if the nurses work overtime, their is no additional cost to the company since the nurses are on salary. In this case, there certainly will be a customer scheduled in the last available time slot (assuming the nurses aren't the ones who do the scheduling).

The optimal schedule depends on the values of μ and C_s. Based on the values of these two parameters, two criteria can be used to eliminate candidates: late-start scheduling and clustering.

3.2 Late-Start Scheduling

It is intuitively obvious that there would be no advantage to scheduling the first appointment any later than the first time unit. This proposition is formally proved in the next chapter. This result eliminates over one half of all schedules from consideration as

being optimal. The number eliminated is given by:

$$\binom{N+k-2}{k-2}$$

The number remaining is:

$$\binom{N+k-2}{k-3}$$

3.3 Clustering

In almost any real-world situation, we would not expect to schedule many appointments at the exact same time if $N < k$. However, depending on the values of the parameters μ and C_s, we can get clusters of any size in the optimal solution. In general, an extremely high value of the overtime cost factor can cause clustering in the early appointments. A very small service rate combined with a relatively low overtime cost factor can cause clustering in the late appointments. Since we are scheduling in a finite time window with discrete appointment times, we cannot be sure that the optimal schedule will not have any component valued at m (i.e., an m-cluster) scheduled in some time slot. We usually can eliminate many schedules which contain specific m-clusters from consideration as being optimal. In order to do this we need an initial feasible solution, which can be a judicious guess. Often a good guess is a schedule which has all of its appointments fairly evenly spread out over the interval . For Example A, we use [1101011] as our initial guess and get a total cost of 4.47. From this guess we can eliminate specific m-clusters.

Obviously, any schedule with an m-cluster that contributes a total expected waiting time cost of more than 4.47 cannot be optimal. A 5-cluster yields expected waiting time of $10/\mu$ time units. This is true because the expected waiting time of the second customer is $1/\mu$, the expected waiting time of the third customer is $2/\mu$, the expected waiting time of the fourth customer is $3/\mu$, etc.. Similarly, a 4-cluster yields expected waiting time of 6 time units. Therefore, no optimal schedule could ever contain either a 4-cluster or 5-cluster.

IV. Foundation

The algorithm is an iterative method for determining the optimal solution. We start by placing all of the appointments in the last time slot. We then move one appointment earlier and check for improvement. If the total cost decreases, this becomes our incumbent upper bound. We prove a theorem which shows that any schedule with appointments scheduled later than the new schedule cannot yield any improvement. We continue the iteration process until we cannot get improvement by moving any one appointment earlier, and then stop the iteration process. Next, we duplicate the process except that we place all appointments in the first time slot and move them later via the iteration process. Again, at each iteration step we know that any schedule with appointments scheduled earlier than the new schedule cannot yield any improvement. We end up with two schedules. No optimal schedule can have an appointment which is scheduled later than its corresponding appointment for the first schedule. Also, no optimal schedule can have an appointment which is scheduled earlier than its corresponding appointment for the second schedule. Clearly, if the two schedules formed by the iteration process are equal, we have the optimal solution. If the two schedules are not equal, we need only consider the schedules "between" the two as candidate schedules. We then enumerate the remaining candidate schedules.

The following proposition shows that every optimal schedule has at least one appointment scheduled in the first time slot:

Proposition 1 Any optimal schedule has at least one appointment scheduled at t_1.

Proof:

Let S_1 be a schedule with its first appointment scheduled later than t_1. Without loss of generality, assume it is scheduled at t_2. Now consider S_2, where each appointment of S_2 is scheduled one unit earlier than the corresponding appointment of S_1. Clearly, $W[S_1] = W[S_2]$ and $O[S_1] > O[S_2]$. Hence S_1 cannot be optimal.

We introduce the following Lemma which is used in the proofs of the upcoming theorems:

Lemma 1 Let T_1 and T_2 be two relative subschedules of m appointments and assume T_1 precedes T_2. Define $P_1(i)$ and $P_2(i)$ as the probabilities that there are i customers in the system for T_1 and T_2 respectively. Then immediately after the m^{th} customer of T_2 enters the system, the following hold:

$$P_1(m) \leq P_2(m).$$

$$P_1(m) + P_1(m-1) \leq P_2(m) + P_2(m-1).$$

$$\vdots$$

$$P_1(m) + P_1(m-1) + \cdots + P_1(1) \leq P_2(m) + P_2(m-1) + \cdots + P_2(1).$$

At the time of the m^{th} customer arrival s_m, the probability that there are m customers in the system is equal to the probability that the first customer is still in service. Similarly, the probability that there are $m - l$ customers in the system is equal to the probability that the first l customers have been served and the $(l+1)^{st}$ customer is still in service.

Define $Q(i)$ as the probability that no more than i customers have been served. For example, $Q(3)$ is the probability that 0,1,2,or 3 customers have been served. Thus, $P_1(m) = Q_1(0)$, $P_1(m) + P_1(m-1) = Q(1)$, $P_1(m) + P_1(m-1) + \cdots + P_1(m-l) = Q(l)$, and $P_1(m) + P_1(m-1) + \cdots + P_1(1) = Q(m-1)$. Clearly, since every customer of T_1 is scheduled no later than its corresponding customer of T_2, the probability that i or fewer customers of T_1 have been served cannot be larger than the corresponding probability for T_2, for any $i = 0, ..., m-1$.

The following theorem and its corollaries compare the amount of increase/decrease in expected time which results from moving an appointment one time slot later or earlier (It is obvious via an inductive argument that the results hold when moving appointments any integral number of time slots also). These theorems are the basis for the foundation of the algorithm, since they guarantee that under certain conditions we can eliminate entire classes of schedules from consideration as being optimal.

Theorem 1 Assume service time follows an r-Erlang distribution with customer service rate μ. Let T_1 and T_2 be two relative subschedules of m appointments and k time slots.

Assume the last appointment of T_1 is scheduled at the same time as the last appointment of T_2, and T_1 precedes T_2. Form T_3 by moving the last appointment of T_1 one time slot later. Form T_4 by moving the last appointment of T_2 one time slot later. Then W[T_1] − W[T_3] < W[T_2] − W[T_4]

Example:

$$T_1 = [2111101] \text{ and } T_2 = [2111011]$$

$$T_3 = [21111001] \text{ and } T_4 = [21110101]$$

Proof:

Consider T_1 and T_3. The expected waiting time of each appointment of T_1 is the same as the expected waiting time of its corresponding appointment of T_3 except for a_m. Similarly, the expected waiting time of each appointment of T_2 is the same as the corresponding appointment of T_4, except for the a_m. Hence, when we look at $W[T_1]-W[T_3]$ and $W[T_2]-W[T_4]$, we need only consider the differences in expected waiting time of the m^{th} appointments of each. To obtain these expected waiting times, we need consider $P_1(i)$ and $P_2(i)$, the probability there are exactly i stages of customer work in the system immediately prior to when customer m enters service.

Clearly, the conditions of Lemma 1 hold. Define $q(i)$ as the probability there are exactly i stages of service in the time interval between the original scheduled time of the m^{th} appointment and the newly scheduled time of the m^{th} appointment. Note these probabilities are identical for T_2 and T_4. Furthermore, by the memoryless property of the exponential distribution, these probabilities are independent of the number in the queue. Since each customer has r exponentially distributed phases of service, the expected waiting times of a_m become:

$$W_1[a_m] = P_1(rm-r)*(rm-r)/r\mu + \cdots + P_1(1)*1/r\mu$$

$$W_3[a_m] = P_1(rm-r)*q(0)*(rm-r)/r\mu + \cdots + P_1(1)*q(rm-r-1)*1/r\mu + P_1(1)*q(0)*1/r\mu$$

$$W_2[a_m] = P_2(rm-r)*(rm-r)/r\mu + \cdots + P_2(1)*1/r\mu$$

$W_4[a_m] = P_2(rm-r)*q(0)*(rm-r)/r\mu+\cdots+P_2(1)*q(rm-r-1)*1/r\mu+P_2(1)*q(0)*1/r\mu$

From these equations, we calculate the difference $(\mathrm{W}[T_2] - \mathrm{W}[T_4]) - (\mathrm{W}[T_1] - \mathrm{W}[T_3]) =$

$$(P_2(rm-r) - P_1(rm-r)) * (1 - (q(0) + \cdots + q(rm-r-1))) * 1/r\mu +$$

$$(P_2(rm-r)+P_2(rm-r-1)-P_1(rm-r)-P_1(rm-r-1))*(1-(q(0)+\cdots+q(rm-r-2)))*1/r\mu$$

$$+(P_2(rm-r) + \cdots + P_2(1) - P_1(rm-r) - \cdots - P_1(1)) * (1 - q(0)) * 1/r\mu$$

which must be a positive number. Therefore

$$(\mathrm{W}[T_2] - \mathrm{W}[T_4]) > (\mathrm{W}[T_1] - \mathrm{W}[T_3])$$

Corollary 1 Let T_1 and T_2 be two perfect relative subschedules of m appointments which are origin subschedules of S_1 and S_2 respectively. S_1 and S_2 are relative schedules of k appointments and S_1=$[T_1|T]$,S_2=$[T_2|T]$ for some T. Assume T_1 precedes T_2. Form S_3 by moving the last appointment of T_1 n time units later (but not passing the succeeding appointment). Form S_4 by moving the last appointment of T_2 n time units later (but not passing the succeeding appointment). Then $O^*[S_3] - O^*[S_1] > O^*[S_4] - O^*[S_2]$, where $O^*[S] = W[a_{m+1}] + \cdots + W[a_k] + O[S]$.

Corollary 2 Let T_1 and T_2 be two perfect relative subschedules of m appointments which are terminating subschedules of S_1 and S_2 respectively. S_1 and S_2 are relative schedules of k appointments and S_1=$[T|T_1]$,S_2=$[T|T_2]$ for some T. Assume T_1 precedes T_2. Form S_3 by moving the last appointment of T (in S_1) n time units later (but not passing the succeeding appointment). Form S_4 by moving the last appointment of T (in S_2) n time units later (but not passing the succeeding appointment). Then $O^*[S_3] - O^*[S_1] > O^*[S_4] - O^*[S_2]$, where $O^*[S] = W[a_{m+1}] + \cdots + W[a_k] + O[S]$.

V. Algorithm

5.1 Methodology

For any problem we now have criteria which allows us to eliminate many candidate schedules. Recall Example A and consider the one schedule which places each appointment at its latest allowable scheduled time. We shall call this schedule the ***latest schedule*** and represent it as L^*. Initially, $L^* = [0000005]$. The late start criteria would require L^* to become [1000004]. After an initial guess of [1101110] to get an upper bound on total expected cost (as seen earlier), the clustering criteria would then imply $L^* = [1000013]$. We will use the initial L_1^* to iteratively select a sequence of schedules $L_i{}^*$ each with the property that no candidate schedule can have any appointment scheduled later that its corresponding position of $L_i{}^*$.

Given the current $L_i{}^*$, consider moving one appointment, say a_i, one unit earlier to form L'. If the total expected cost of L' is less than the total expected cost of $L_i{}^*$, then clearly $L_i{}^*$ is no longer a candidate. Furthermore, if any terminating subschedule which begins with a_{i+1} replaces the corresponding subschedule of L^*, moving the i^{th} appointment will result in a schedule with higher total expected cost than that of L'. Remember, by moving a_i one unit earlier, the expected waiting time of a_i will increase, and O^* will decrease. This increase will be the same in both cases, but by Corollary 2, the decrease in O^* is less than between $L_i{}^*$ and L'. Hence no candidate schedule can have any appointment scheduled later than where it is scheduled in L'.

If we keep iterating the above process, we will get to a point where moving any appointment one unit earlier will cause total expected cost to increase. Therefore, we have determined the latest possible time at which any appointment can be scheduled. Next, we repeat the process. We will use an initial L_1^{**} to iteratively select a sequence of schedules $L_i{}^{**}$ each with the property that no candidate schedule can have any appointment scheduled earlier that its corresponding position of $L_i{}^{**}$.

Again, given the current $L_i{}^{**}$, consider moving one appointment, say a_i, one unit later to form L'. If the total expected cost of L' is less than the total expected cost of $L_i{}^{**}$, then clearly $L_i{}^{**}$ is no longer a candidate. Furthermore, if any terminating subschedule which

ends with a_{i+1} replaces the corresponding subschedule of L^{**}, moving the i^{th} appointment will result in a schedule with higher total expected cost than that of L'. By moving a_i one unit later, the expected waiting time of a_i will decrease, and O^* will increase. This decrease will be less for any preceding subschedule and by Corollary 2, the increase in O^* will be more when the the initial subschedule is a preceding subschedule. Hence no candidate schedule can have any appointment scheduled earlier than where it is scheduled in L'. We continue this procedure until we satisfy our stopping criteria.

5.2 Stopping Criteria

When the first iteration phase reaches the point where no improvement is achieved by moving any appointment one time slot earlier, we then begin the second iteration phase. If the same schedule is the solution to both iterations, then we clearly have the optimal schedule since each appointment is scheduled at both its earliest and latest possible times. If the two schedules are different, then we must evaluate all schedules which have appointments that lie between the latest possible and earliest possible scheduled times. In most of the cases we have evaluated, the iteration processes lead to the optimal solution. In the cases where they were not the same, the two schedules differed by at most 8 appointments (for a problem with 25 appointments and 32 time slots). Moreover, for each problem we have tested, one of the two iteration phase schedules has been the optimal one. In general, if the two iteration phase schedules differ by m appointments, there will be no more than 2^m remaining candidate schedules to consider. It is important to note that if a situation requires an immediate solution, one iteration phase will most certainly give a schedule in a timely manner which is close to the optimal one.

There are several approaches we can take in the iteration processes, since we can move up to $N-1$ appointments of the current incumbent bound ($L_i{}^*$). For ease in the coding of the algorithm, we have decided to move each one and then form $L_{i+1}{}^*$ by moving the one which results in the best improvement. Another approach would be to move latest (earliest) appointment and see if it yields improvement. If it does, use that schedule for $L_{i+1}{}^*$, if not, move the next latest (earliest) appointment, etc... The former approach would result in the evaluation of more schedules, but the latter approach would require more iterations

of checking whether or not you get improvement by moving one appointment. Since either method only requires the evaluation of a small subset of the original candidate schedules, we feel that the first method is efficient enough and also gives more sensitivity analysis if the schedules from the last few iterations are observed.

Using our first method, we could have at most $(N-1)*(k-1)$ iterations for each phase, and in each iteration we will evaluate at most $(k-1)$ schedules. Hence, the maximum number of schedules to calculate is $2*(k-1)^2*(N-1)$. In practice, we will always calculate many fewer. The only way one could achieve a number near this maximum would be if the expected waiting cost is practically zero. Nevertheless, if for $N = 25$, and $k = 32$, the number of feasible schedules is over $5.57*10^{15}$, while the algorithm could theoretically require the calculation of at most 23,064 schedules for each phase. If the two phases are solved simultaneously, we would start out with the same initial upper bound for each one. If they are done in succession, we will have a very good starting point from the first phase solution and hence can use it for the second phase. Generally, the phase where we move appointments to the right converges much more rapidly than when move appointments to the left. Thus, if these phases are done in succession, it would be better to get the earliest possible appointments and use the total expected cost to get an initial guess for the phase of the algorithm which determines the latest possible appointments.

If the two phases of the algorithm yield different solutions, then we must enumerate the remaining schedules in order to determine if there is a better schedule than our best so far. In all of the examples we investigated, we found at most eight apppointments that were different between the two phases (in the cost measured as overtime case). Thus, complete enumeration would require the calculation of at most $2^8 = 256$ additional schedules. However, it is reasonable to assume that an example could occur where the two solutions differed by many more appointments. If the number of schedules remaining to be calculated is large, then we would have to use some branch and bound or implicit enumeration technique to eliminate large numbers of candidate schedules quickly. We could use our best solution as an initial lower bound for the branch and bound technique developed by Liao (9). In fact, our initial lower bound may indeed be achieved more quickly than with the

dynamic programming/branch and bound techniques employed by Liao. Moreover, one of his recommendations is for a better technique for establishing an initial lower bound.

An additional benefit of the algorithm is that it can be used to determine potential gain by using a smaller time increment for the schedule. For example, if $N = 5$ and $k = 7$, we might want to know if there is any advantage to doubling the number of time slots but making each one half the length of the original ones. This reiteration of the algorithm could require the calculation of at most $(2k - 1)^2 * (N - 1)$ schedules for each iteration phase if run from the start. However, we can use information from the incumbent optimal solution to establish a good upper bound, and the incumbent solution has been established with the new time slots, we move each appointment two new units to the later (earlier). Clearly, this schedule satisfies all requirements for L^* in the algorithm. We note that it will be necessary to determine the subdivision increments based on the prime factorization of the time interval.

It is important to note that the maximum number of schedules required to calculate at each step grows only linearly. In fact, at each step we need calculate no more than $2 * l * N$ schedules, where l =the number of subdivisions of the previous time slot. For this reason, the algorithm is also an efficient method to get an approximation to the continuous time versions of the problem. If we halve our interval at each iteration step, we will reach machine zero at a relatively small number of iterations. Since the complexity of algorithm only grows linearly, we can reach an approximation to the continuous-time solution with the calculation of only a relatively small number of candidate schedules.

There are two numerical reasons why the algorithm only gives an approximate solution to the continuous problem. First, as we shrink the size of the time slots, there is a greater liklihood that the two iteration steps will yield solutions which differ by many appointments, thus making enumeration impractical. Additionally, since we are exponentiating extremely small numbers, the two iteration processes will have built up different amounts of machine error, again causing the two solutions to differ by many time slots.

5.3 Sample Problem

Consider our Example A. For these values of the parameters, we have seen that we can quickly eliminate 4 and 5-clusters from consideration. Therefore, we start out with [1000013] as L^*. We then check the value of the schedule formed by moving each appointment one unit left. We will accept the schedule which results in the lowest total cost. The following lists the sequence of schedules which were chosen:

[1000013] Total Expected Cost: 13.68

[1000103] Total Expected Cost: 12.10

[1000112] Total Expected Cost: 9.54

[1001012] Total Expected Cost: 8.38

[1001102] Total Expected Cost: 7.61

[1001111] Total Expected Cost: 6.34

[1010111] Total Expected Cost: 5.47

[1011011] Total Expected Cost: 5.02

[1101011] Total Expected Cost: 4.74

[1101101] Total Expected Cost: 4.42

[1110101] Total Expected Cost: 4.35

[1110110] Total Expected Cost: 4.10

Next, we perform the second iteration process.

The following are the phase two results:

[3110000] Total Expected Cost: 8.27

[3101000] Total Expected Cost: 7.53

[2201000] Total Expected Cost: 6.76

[2111000] Total Expected Cost: 5.96

[2101100] Total Expected Cost: 5.06

[2011100] Total Expected Cost: 4.79

[1111100] Total Expected Cost: 4.48

[1111010] Total Expected Cost: 4.17

[1110110] Total Expected Cost: 4.10

Since the two iteration phases yield identical schedules, we have the optimal solution.

Out of the 462 possible schedules, we only calculated 66 in order to get the phase one solution.

VI. Examples

We present some sample problems with a variety of parameters. These examples illustrate how the algorithm performs with different distributions and various combinations of numbers of appointments and numbers of time slots.

6.1 k=15, N=40

The following two examples represent problems where there are many more time slots than appointments.

CASE I: $\mu = 1$, C = 3, r = 1

Both iteration processes yield the same schedule:

[1010010100100101001001010010010100100100]

This is the optimal solution. The objective function value is 1.69 . Out of the $8.65 * 10^{12}$ possible schedules, 7686 schedules were evaluated.

CASE II: $\mu = 1$, C = 3, r = 2

The latest candidate schedule is: [1010010100100101001001010010010100100100] The objective function value is: .43659

The earliest candidate schedule is: [1010010010100100101001001010010010100100] The objective function value is: .43646

The two schedules are identical except for the 4^{th}, 7^{th}, 10^{th}, and 13^{th} appointments. Therefore, we only need to evaluate the schedules which result from moving combinations of these appointments. Moreover, we don't need to check the schedules formed by moving exactly one appointment, exactly three appointments or exactly four appointments. We have already checked the ones formed by moving one appointment. Moving three appointments from one of the two candidates is equivalent to moving one on the other. Moving all four would give the other candidate. Hence, we only need to evaluate all schedules formed by moving exactly two appointments, and there are 6 of these schedules.

None of these schedules we enumerated had a lower total expected cost than either of our schedules formed via the iteration processes. Thus, the optimal schedule is the earliest

candidate schedule above, and the objective function value is .43646. 8064 schedules were evaluated in the iteration processes, plus 6 in the enumeration process for a total of 8070.

6.2 k=20, N=16

The following two examples represent problems where there are about the same number of time slots as appointments.

CASE I: $\mu = 1$, $C = 0$, $r = 2$

Both iteration processes yield the same schedule: [11101101101101110112] The objective function value is 7.11. This is the optimal solution. Out of the $4.06 * 10^9$ possible schedules, 4275 schedules were evaluated.

CASE II: $\mu = 1$, $C = 50$, $r = 3$

Both iteration processes yield the same schedule: [11111111101111011100] The objective function value is 15.07. This is the optimal solution. 4275 schedules were evaluated.

6.3 k=32, N=25

The following two examples represent a larger scale problem with many appointments and many time slots.

CASE I: $\mu = 1$, $C = 25$, $r = 1$

The latest candidate schedule is: [21111011110111101111011101110100] The objective function value is: 46.58

The earliest candidate schedule is: [21111101111011110111011101110100] The objective function value is: 46.60

These two schedules are identical except for the 6^{th}, 10^{th}, and 14^{th} appointments. We already know the schedules formed by moving one appointment are not candidates, hence the schedules formed by moving two appointments are not candidates either, since it is identical to a schedule formed by moving one appointment in the other iteration. Clearly moving all three produces the other iteration schedule. Thus, the optimal schedule is the

latest candidate schedule above, and the objective function value is 46.58. 23653 schedules were evaluated.

CASE II: $\mu = 1$, C = 100, r = 2

Both iteration processes yield the same schedule: [211110111111011110111110111011000] The objective function value is 32.73. This is the optimal solution. 24087 schedules were evaluated.

6.4 Server Shuts Down, k=20, N=16

In this situation, server time is measured as the length of time from when the finite time horizon starts until the expected time that the last customer departs service. Service cost is measured as the cost per unit multiplied by the expected server time. Again, we need only consider the ratio of customer waiting cost to server cost. We first check an example with server cost is 0 in order to see we get the same result as when server cost is measured as overtime, and server cost equals 0.

CASE I: $\mu = 1$, C = 0, r = 2

Both iteration processes yield the same schedule: [11101101101101110112] The objective function value is 7.11. This is the optimal solution. Out of the $4.06 * 10^9$ possible schedules, 4275 schedules were evaluated.

CASE II: $\mu = 1$, C = 5, r = 2

As expected, our answer is different from the overtime cost example. The latest candidate schedule is: [11111110111110111100]. The objective function value is 111.90

The earliest candidate schedule is: [11111111111111110000]. The objective function value is: 112.66

If we assume there will be no two cluster, there 50 additional schedules to calculate. If not, there are 2^{11} to consider. After enumerating all 2^{11} schedules, we see that the latest candidate schedule is the optimal one.

6.5 *Continuous Case, k=7, N=5, μ=1, C_s=4*

We use the algorithm to get an approximation to the continuous problem of our Example A. The program halved the given time slots and solved with the smaller intervals 308 times.

The latest candidate schedule is :[0, .12236, .32081, .52981, .74116]

The earliest candidate schedule is :[0, .12234, .32079, .52972, .74104]

The objective function value for both is: 3.98251, and was identical for all 13 decimal places of each answer. We assume machine error will be so varied between the two iteration phases that getting a very precise approximation will be difficult. However, we have an approximation correct to 3 decimal places. Also, we have a small neighborhood where we know each appointment must be scheduled. Hence the algorithm is useful in approximating the solution to the continuous problem. Each iteration phase stopped after there were over 10^{308} subintervals created.

VII. Conclusions and Recommendations

Our algorithm is a very efficent method for solving the optimal control of arrivals problem. The algorithm solves the problem for the case where server cost is measured as expected overtime cost, as well as the case where server time is measured as the difference in time between when the server starts and when the server shuts down. The only difference will be in the cost factor and how the server cost is measured.

The algorithm can likely be extended to the following cases:

- Multiple parallel servers with r-Erlang distributed service time.
- Multiple servers in sequence each with r-Erlang distributed service time.
- Several classes of customers in the system, each class having different k-Erlang waiting time distributions and waiting cost structures.
- All of the above cases where service time follows a general distribution.

The theorems appear extendable to most nice distributions. Hence it seems likely that the algorithm in the single server example would work with any unimodal distribution for service time. Also, the methodology appears to be fully extendable to to the cases of multiple servers and classes of customers, although the computer record keeping may slow down the efficiency, particularly when there are multiple classes of customers. Additionally, the algorithm appears to work for approximating continuous-time versions of the aforementioned problems as well. Extensions of this research effort are needed to verify the applicability of the algorithm to the additional cases.

Appendix A. Alternate Proofs

A.1 Alternate Proof of Theorem 1

Theorem 1 Let T_1 and T_2 be two relative subschedules of m appointments and k time slots, and assume T_1 precedes T_2. Form T_3 by moving the last appointment of T_1 n time units later. Form T_4 by moving the last appointment of T_2 n time units later. Then $\mathrm{W}[T_1] - \mathrm{W}[T_3] < \mathrm{W}[T_2] - \mathrm{W}[T_4]$

Proof: We know that $W_1[a_m] < W_2[a_m]$. The total time to complete service, x_1, for each customer already in the queue for T_1 is distributed according to a Gamma distribution with parameters $r_1 = W_1[a_m]$ and $\theta = \mu$. Similarly, the total time to complete service, x_2, for each customer already in the queue for T_2 is distributed according to a Gamma distribution with parameters $r_2 = W_2[a_m]$ and $\theta = \mu$. It is a fact that if two Gamma distributions with cumulative distribution functions Γ_1 and Γ_2 have the same θ parameter and $r_1 < r_2$, then $\Gamma_2(t) > \Gamma_1(t)$ for all $t \geq 0$ over any time interval. Therefore the difference in the expected number of customers served by moving the last appointment of T_1 later is always less than the difference in the expected number of customers served by moving the last appointment of T_2 later.

A.2 Alternate Proof of Corollary 1

Corollary 1 Let T_1 and T_2 be two perfect relative subschedules of m appointments which are origin subschedules of S_1 and S_2 respectively. S_1 and S_2 are relative schedules of k appointments and S_1=$[T_1|T]$,S_2=$[T_2|T]$ for some T. Assume T_1 precedes T_2. Form S_3 by moving the last appointment of T_1 n time units later (but not passing the proceeding appointment). Form S_4 by moving the last appointment of T_2 n time units later (but not passing the proceeding appointment). Then $O^*[S_3] - O^*[S_1] > O^*[S_4] - O^*[S_2]$ where $O^*[S] = W[a_{m+1}] + \cdots + W[a_k] + O[S]$.

Proof: As seen in the proof of Theorem 1 the total time x_1 to complete service for each customer already in the queue for T_1 is distributed according to a Gamma distribution with parameters $r_1 = W_1[a_m]$ and $\theta = \mu$. Similarly, the total time x_2 to complete service for each customer already in the queue for T_2 is distributed according to a Gamma distribution with

parameters $r_1 = W_1[a_m]$ and $\theta = \mu$, where $\Gamma_2(t) > \Gamma_1(t)$ for all $t \geq 0$. Hence $E[x_2] > E[x_1]$ over any time interval. Consider the cumulative distribution F_1 of the number not served in T_1 over any time interval $(0, t)$, which will be the same as $\overline{\Gamma_1(t)}$. Similarly, the distribution F_2 of the number not served in T_2 over that same time interval, will be equal to $\overline{\Gamma_2(t)}$. Clearly, F_1 is always greater than or equal to F_2, which implies the difference in expected waiting time of the $(m+1)^{st}$ customer after moving the m^{th} customer of T_1 later is greater then the difference in expected waiting time of the $(m+1)^{st}$ customer after moving the m^{th} customer of T_2 later. Since T_1 precedes T_2, and $W_1[a_{m+i}] < W_2[a_{m+i}]$, $i = 1, \ldots, k - m$ the theorem will hold for every appointment succeeding a_m, as well as for expected overtime.

Appendix B. Computer Code

```
*  THIS CODE FINDS EARLIEST CANDIDATE SCHEDULE--- SERVICE = r-ERLANG
* SERVER COST MEASURED AS OVERTIME
         REAL*8 X(200,200),Y(200,200),P(200,200),Q(200,200),W(200)
         REAL*8 U,SUM,SUM1,SUM2, GG
         REAL*8 WAIT,STORE,C,OT,TEMP,FACT,BEST,VAL,OLDBEST,TOTAL,N
         REAL*8 NAPT,NSLT,CHECK,CUSMU,STAGES,BWAIT,BOT
         INTEGER I,A,J,D,L,K,M,Z,ARF
         OPEN (UNIT=2, FILE='algol.out')
         BEST=10000000.
         OLDBEST=1000000.
         ARF=0
* INPUT PARAMETERS
         N=25.
         NSLT=32.
         C=0.
         STAGES=2.
         CUSMU= 1.
         WRITE(2,*) STAGES, '-Erlang' ,' U=', CUSMU, N, '  CUSTOMERS'
         WRITE(2,*) 'Cost Factor =   ', C,'server cost is overtime'
         WRITE(2,*) '# of slots =    ', NSLT
         U=CUSMU*STAGES
         TOTAL=N+1
         NAPT=N*STAGES
         DO 1 F=1,NAPT+1.
             DO 2 G=1,F
              P(F,G)=0.
              Q(F,G)=0.
2             CONTINUE
1          CONTINUE
* INITIALIZE FIRST SCHEDULE
         DO 3 G=1, TOTAL
             DO  434 CC=1, NAPT-1.
                X(G,CC)=0.
434          CONTINUE
             X(G,NAPT)=NSLT
3        CONTINUE
130      DO 2000 A=1,N-1.
             DO 11 B=1,NAPT
                 Y(A,B)=X(A,B)*U
11           CONTINUE
         Z=0
*        next calculate the P probabilities
         DO 201 I=1,NAPT
             Z=Z+1
             SUM=0.
             FACT=1.
*
             DO 101 J=1,I
                P(I,J)=EXP(-Y(A,Z))*((Y(A,Z)**(J-1))/FACT)
                SUM=SUM+P(I,J)
                FACT=FACT*(J)
101          CONTINUE
            P(I,I+1)=1.-SUM
201      CONTINUE
*
*        calculate the Q probabilities
         Q(1,2)=P(1,1)
         Q(1,1)=1-Q(1,2)
         DO 10 K=2,NAPT
           SUM1=0.
           DO 12 L=1,K
              SUM2=0.
              TEMP=0.
              D=1
              DO 14 M=L, K
                TEMP=P(K,D)*Q(K-1,M)
```

```
                SUM2=SUM2+TEMP
                D=D+1
14            CONTINUE
         Q(K,L+1)=SUM2
         SUM1=SUM1+SUM2
12       CONTINUE
         Q(K,1)=1.-SUM1
10     CONTINUE
       DO 7 R=1,NAPT
         W(R)=0.
7      CONTINUE
       DO 33 E=1,NAPT
          STORE=0.
          DO 44 F=2,E+1
             STORE=Q(E,F)*(F-1)/U
             W(E)=W(E)+STORE
44        CONTINUE
33     CONTINUE
       WAIT=0.
       OT=0.
       CHECK=1.
       DO 55 G=1,NAPT-1
          DO 555 GG=1.,NAPT
             IF ((CHECK/STAGES).EQ. GG) WAIT=W(G)+WAIT
555       CONTINUE
          CHECK=CHECK+1.
55     CONTINUE
       OT=W(NAPT)*C
        VAL=WAIT+OT
        IF ((VAL.GE.BEST) .OR. (VAL.EQ.0)) GOTO 2000
        BEST=VAL
        BWAIT=WAIT
        BOT=OT
        DO 241 MM=1,NAPT
           X(N+1,MM)=X(A,MM)
241     CONTINUE
2000   CONTINUE
       IF (BEST.GE.OLDBEST) PRINT*, 'hallelula'
       IF (BEST.GE.OLDBEST) GOTO 131
       DO 30 GG=1, N
           X(GG,1.)=X(N+1.,1.)
           DO 40 HH=2,NAPT-1
               X(GG,HH)=X(N+1.,HH)
40         CONTINUE
           X(GG,NAPT)=X(N+1.,NAPT)
30      CONTINUE
*
        DO 1747 PP=1,N-1.
           DO 888 JJ=1,N-1.
           IF (X(PP,PP*STAGES+JJ*STAGES) .EQ. 0.) GOTO 888
        IF ( X(PP, PP*STAGES+JJ*STAGES) .EQ. 1) ARF=3+ARF
        IF ((PP*STAGES+JJ*STAGES) .EQ. NAPT) ARF=3+ARF
        IF (ARF .EQ. 6) GOTO 2468
           X(PP,PP*STAGES+JJ*STAGES)=X(PP,PP*STAGES+JJ*STAGES)-1.
         X(PP,PP*STAGES)=X(PP,PP*STAGES)+1.
2468     GOTO 1746
888     CONTINUE
1746    ARF=0
1747    CONTINUE
       PRINT*, 'dabestis', BEST
       WRITE(2,*)  'dabeast', BEST
       OLDBEST=BEST
       GOTO  130
131    PRINT*, 'youreatheend'
       PRINT*, 'verybestis', BEST
       PRINT*, (X(N+1,TT),TT=1,NAPT)
```

```
      WRITE(2,*)  (X(N+1,RR),RR=1,NAPT)
      WRITE(2,*) BEST, '= theverybest', 'WAIT=', BWAIT, 'OT=', BOT
789   FORMAT(F5.1)
      STOP
      END
```

```
              D=1
              DO 14 M=L, K
                TEMP=P(K,D)*Q(K-1,M)
                SUM2=SUM2+TEMP
                D=D+1
14            CONTINUE
           Q(K,L+1)=SUM2
           SUM1=SUM1+SUM2
12         CONTINUE
           Q(K,1)=1.-SUM1
10       CONTINUE
         DO 7 R=1,NAPT
           W(R)=0.
7        CONTINUE
         DO 33 E=1,NAPT
            STORE=0.
            DO 44 F=2,E+1
               STORE=Q(E,F)*(F-1)/U
               W(E)=W(E)+STORE
44          CONTINUE
33       CONTINUE
         WAIT=0.
         OT=0.
         CHECK=1.
         DO 55 G=1,NAPT-1
            DO 555 GG=1.,NAPT
               IF ((CHECK/STAGES).EQ. GG) WAIT=W(G)+WAIT
555         CONTINUE
            CHECK=CHECK+1.
55       CONTINUE
         OT=W(NAPT)*C
          VAL=WAIT+OT
          IF ((VAL.GE.BEST) .OR. (VAL.EQ.0)) GOTO 2000
          BEST=VAL
          BWAIT=WAIT
          BOT=OT
          DO 241 MM=1,NAPT
             X(N+1,MM)=X(A,MM)
241       CONTINUE
2000     CONTINUE
*         IF (BEST.GE.OLDBEST) PRINT*, 'hallelula'
         IF (BEST.GE.OLDBEST) GOTO 131
* INITIALIZE ALL SCHEDULES TO CURRENT BEST
         DO 30 GG=1, N
             X(GG,1.)=X(N+1.,1.)
             DO 40 HH=2,NAPT-1
                 X(GG,HH)=X(N+1.,HH)
40           CONTINUE
             X(GG,NAPT)=X(N+1.,NAPT)
30        CONTINUE
         DO 747 PP=1,N-1.
* MOVE THE APPOINMEMNT
         IF (X(N+1.,STAGES*PP) .EQ. 0.)  GOTO 747
             X(PP,PP*STAGES)=X(PP,PP*STAGES)-1.
             X(PP,PP*STAGES+STAGES)=X(PP,PP*STAGES+STAGES)+1.
747      CONTINUE
*         PRINT*, 'bestis', BEST
*         WRITE(2,*)  'best', BEST
         OLDBEST=BEST
         GOTO  130
131      PRINT*, 'youreatheend'
         PRINT*, 'verybestis', BEST
         PRINT*, (X(N+1,TT),TT=1,NAPT)
         WRITE(2,*) (X(N+1,RR),RR=1,NAPT)
         WRITE(2,*) BEST, '= theverybest', 'WAIT=', BWAIT, 'OT=', BOT
789      FORMAT(F5.1)
```

```
* THIS PROGRAM FINDS THE LATEST CANDIDATE SCHEDULE FOR r-ERLANG
* DISTRIBUTIONS --- SERVER COST MEASURED AS OVERTIME
         REAL*8 X(200,200),Y(200,200),P(200,200),Q(200,200),W(200)
         REAL*8 U,SUM,SUM1,SUM2
         REAL*8 WAIT,STORE,C,OT,TEMP,FACT,BEST,VAL,OLDBEST,TOTAL,N
         REAL*8 NAPT,NSLT,CHECK,CUSMU,STAGES,BWAIT,BOT
         INTEGER I,A,J,D,L,K,M,Z
         OPEN (UNIT=2, FILE='algo.out')
         BEST=10000000.
         OLDBEST=10000000.
*       INPUT PARAMETERS
         N=25.
         NSLT=32.
         C=0.
         STAGES=2.
         CUSMU= 1.
         WRITE(2,*) STAGES, '-Erlang' ,' U=', CUSMU, N, '  CUSTOMERS'
         WRITE(2,*) 'Cost Factor =  ', C,'server cost is overtime'
         WRITE(2,*) '# of slots =   ', NSLT
         U=CUSMU*STAGES
         TOTAL=N+1
         NAPT=N*STAGES
         DO 1 F=1,NAPT+1.
             DO 2 G=1,F
              P(F,G)=0.
              Q(F,G)=0.
2             CONTINUE
1         CONTINUE
*        INITIALIZE THE FIRST SCHEDULE
         DO 3 G=1, TOTAL
             DO  434 CC=1, STAGES-1.
                X(G,CC)=0.
434          CONTINUE
             X(G,STAGES)=NSLT-1.
             DO 4 H=STAGES+1.,NAPT-1.
             X(G,H)=0.
4            CONTINUE
             X(G,NAPT)=1.
3        CONTINUE
130      DO 2000 A=1,N-1.
             DO 11 B=1,NAPT
                 Y(A,B)=X(A,B)*U
11           CONTINUE
         Z=0
*        calculate the P probabilities for each schedule
         DO 201 I=1,NAPT
             Z=Z+1
             SUM=0.
             FACT=1.
*
             DO 101 J=1,I
                P(I,J)=EXP(-Y(A,Z))*((Y(A,Z)**(J-1))/FACT)
                SUM=SUM+P(I,J)
                FACT=FACT*(J)
101          CONTINUE
            P(I,I+1)=1.-SUM
201      CONTINUE
*
*        calculate the Q probabilities=P(i services)
         Q(1,2)=P(1,1)
         Q(1,1)=1-Q(1,2)
         DO 10 K=2,NAPT
           SUM1=0.
           DO 12 L=1,K
              SUM2=0.
              TEMP=0.
```

```
STOP
END
```

```
*·THIS CODE FINDS EARLIEST CANDIDATE SCHEDULE--- SERVICE = r-ERLANG
* SERVER COST MEASURED AS TIME UNTIL SERVER SHUTS DOWN
         REAL*8 X(200,200),Y(200,200),P(200,200),Q(200,200),W(200)
         REAL*8 U,SUM,SUM1,SUM2, GG
         REAL*8 WAIT,STORE,C,OT,TEMP,FACT,BEST,VAL,OLDBEST,TOTAL,N
         REAL*8 NAPT,NSLT,CHECK,CUSMU,STAGES,BWAIT,BOT
         INTEGER I,A,J,D,L,K,M,Z,ARF,PLACE
         OPEN (UNIT=2, FILE='parm1.out')
         BEST=10000000.
         OLDBEST=1000000.
         PLACE=0
         ARF=0
         N=16.
         NSLT=20.
         C=5.
         STAGES=2.
         CUSMU= 1.
         WRITE(2,*) STAGES, '-Erlang' ,' U=', CUSMU, N, '  CUSTOMERS'
         WRITE(2,*) 'Cost Factor =  ', C,'server cost is overtime'
         WRITE(2,*) '# of slots =   ', NSLT
         U=CUSMU*STAGES
         TOTAL=N+1
         NAPT=N*STAGES
         DO 1 F=1,NAPT+1.
             DO 2 G=1,F
              P(F,G)=0.
              Q(F,G)=0.
2             CONTINUE
1         CONTINUE
         DO 3 G=1, TOTAL
             DO  434 CC=1, NAPT-1.
                 X(G,CC)=0.
434          CONTINUE
             X(G,NAPT)=NSLT
3        CONTINUE
130      DO 2000 A=1,N-1.
             DO 11 B=1,NAPT
                 Y(A,B)=X(A,B)*U
11           CONTINUE
*         PRINT*, 'i initialized everything'
         Z=0
*        next calculate the P probabilities
         DO 201 I=1,NAPT
             Z=Z+1
             SUM=0.
             FACT=1.
*
             DO 101 J=1,I
                P(I,J)=EXP(-Y(A,Z))*((Y(A,Z)**(J-1))/FACT)
                SUM=SUM+P(I,J)
                FACT=FACT*(J)
101          CONTINUE
            P(I,I+1)=1.-SUM
201      CONTINUE
*
*        calculate the Q probabilities
         Q(1,2)=P(1,1)
         Q(1,1)=1-Q(1,2)
         DO 10 K=2,NAPT
           SUM1=0.
           DO 12 L=1,K
              SUM2=0.
              TEMP=0.
              D=1
              DO 14 M=L, K
                 TEMP=P(K,D)*Q(K-1,M)
```

```
                SUM2=SUM2+TEMP
                D=D+1
14            CONTINUE
          Q(K,L+1)=SUM2
          SUM1=SUM1+SUM2
12        CONTINUE
          Q(K,1)=1.-SUM1
10      CONTINUE
        DO 7 R=1,NAPT
          W(R)=0.
7       CONTINUE
        DO 33 E=1,NAPT
           STORE=0.
           DO 44 F=2,E+1
              STORE=Q(E,F)*(F-1)/U
              W(E)=W(E)+STORE
44         CONTINUE
33      CONTINUE
        WAIT=0.
        OT=0.
        CHECK=1.
        DO 55 G=1,NAPT-1
           DO 555 GG=1.,NAPT
              IF ((CHECK/STAGES).EQ. GG) WAIT=W(G)+WAIT
555        CONTINUE
           CHECK=CHECK+1.
55      CONTINUE
*
         IF (X(A,NAPT).EQ. 0.) PLACE=NSLT
          DO 929 SS=1, NSLT
            IF (X(A,NAPT).EQ.SS) PLACE= NSLT-X(A,NAPT)+1.
929      CONTINUE
930        OT=(W(NAPT-1.)+PLACE+1./U)*C
         VAL=WAIT+OT
         IF ((VAL.GE.BEST) .OR. (VAL.EQ.0)) GOTO 2000
         BPLACE=PLACE
         BEST=VAL
         BWAIT=WAIT
         BOT=OT
         DO 241 MM=1,NAPT
            X(N+1,MM)=X(A,MM)
241      CONTINUE
2000    CONTINUE
        IF (BEST.GE.OLDBEST) PRINT*, 'hallelula'
        IF (BEST.GE.OLDBEST) GOTO 131
        DO 30 GG=1, N
            X(GG,1.)=X(N+1.,1.)
            DO 40 HH=2,NAPT-1
                X(GG,HH)=X(N+1.,HH)
40          CONTINUE
            X(GG,NAPT)=X(N+1.,NAPT)
30       CONTINUE
*
         DO 1747 PP=1,N-1.
            DO 888 JJ=1,N-1.
            IF (X(PP,PP*STAGES+JJ*STAGES) .EQ. 0.) GOTO 888
         IF ( X(PP, PP*STAGES+JJ*STAGES) .EQ. 1) ARF=3+ARF
         IF ((PP*STAGES+JJ*STAGES) .EQ. NAPT) ARF=3+ARF
         IF (ARF .EQ. 6) GOTO 2468
            X(PP,PP*STAGES+JJ*STAGES)=X(PP,PP*STAGES+JJ*STAGES)-1.
          X(PP,PP*STAGES)=X(PP,PP*STAGES)+1.
2468      GOTO 1746
888      CONTINUE
1746     ARF=0
1747     CONTINUE
*
```

```
      PRINT*, 'bestis', BEST
      WRITE(2,*)  'beast', BEST
      OLDBEST=BEST
      GOTO  130
131   PRINT*, 'youreatheend'
      PRINT*, 'verybestis', BEST
      PRINT*, (X(N+1,TT),TT=1,NAPT)
      WRITE(2,*) (X(N+1,RR),RR=1,NAPT)
      WRITE(2,*) BEST, '= theverybest', 'WAIT=', BWAIT, 'OT=', BOT
789   FORMAT(F5.1)
      STOP
      END
```

```
* THIS CODE FINDS LATEST CANDIDATE SCHEDULE--- SERVICE = r-ERLANG
* SERVER COST MEASURED AS TIME UNTIL SERVER SHUTS DOWN
        REAL*8 X(200,200),Y(200,200),P(200,200),Q(200,200),W(200)
        REAL*8 U,SUM,SUM1,SUM2,PLACE,CUSMU,STAGES,GG
        REAL*8 WAIT,STORE,C,OT,TEMP,FACT,BEST,VAL,OLDBEST,TOTAL,N
        REAL*8 NAPT,NSLT,BWAIT,BOT,SS,BPLACE
        INTEGER I,A,J,D,L,K,M,Z
        OPEN (UNIT=2, FILE='3exp2532srv.out')
        BEST=100000000.
        OLDBEST=100000000.
        PLACE=0.
        N=5.
        NSLT=7.
        STAGES=1.
        CUSMU=1.
        U=CUSMU*STAGES
        C=4.
        WRITE(2,*) STAGES, '-Erlang' ,' U=', CUSMU, N, '  CUSTOMERS'
        WRITE(2,*) 'Cost Factor =  ', C,'server cost '
        WRITE(2,*) '# of slots =    ', NSLT
*      HERE's where i'd go back
        TOTAL=N+1
        NAPT=N*STAGES
        DO 1 F=1,N*STAGES+1.
            DO 2 G=1,F
             P(F,G)=0.
             Q(F,G)=0.
2            CONTINUE
1        CONTINUE
        DO 3 G=1, TOTAL
            DO  434 CC=1, STAGES-1.
               X(G,CC)=0.
434         CONTINUE
            X(G,STAGES)=NSLT-1.
            DO 4 H=STAGES+1.,NAPT-1.
            X(G,H)=0.
*            PRINT*, 'X(G,H)', X(G,H)
4           CONTINUE
            X(G,NAPT)=1.
*            PRINT*,'X(G,NAPT)=' , X(G,NAPT)
3       CONTINUE
130     DO 2000 A=1,N-1
            DO 11 B=1,NAPT
                Y(A,B)=X(A,B)*U
11          CONTINUE
*        PRINT*, 'i initialized everything'
        Z=0
*        PRINT*, 'calculate p'
*       next calculate the P probabilities
        DO 201 I=1,NAPT
            Z=Z+1
            SUM=0.
            FACT=1.
*
            DO 101 J=1,I
               P(I,J)=EXP(-Y(A,Z))*((Y(A,Z)**(J-1))/FACT)
               SUM=SUM+P(I,J)
               FACT=FACT*(J)
101         CONTINUE
           P(I,I+1)=1.-SUM
201     CONTINUE
*
*        PRINT*, ' i got to q'
*       calculate the Q probabilities
        Q(1,2)=P(1,1)
        Q(1,1)=1-Q(1,2)
```

```
          DO 10 K=2,NAPT
            SUM1=0.
            DO 12 L=1,K
               SUM2=0.
               TEMP=0.
               D=1
               DO 14 M=L, K
                 TEMP=P(K,D)*Q(K-1,M)
                 SUM2=SUM2+TEMP
                 D=D+1
14             CONTINUE
            Q(K,L+1)=SUM2
            SUM1=SUM1+SUM2
12          CONTINUE
            Q(K,1)=1.-SUM1
10        CONTINUE
          DO 7 R=1,NAPT
            W(R)=0.
7         CONTINUE
          DO 33 E=1,NAPT
             STORE=0.
             DO 44 F=2,E+1
                STORE=Q(E,F)*(F-1)/U
                W(E)=W(E)+STORE
44           CONTINUE
33        CONTINUE
          WAIT=0.
          OT=0.
          CHECK=1.
          DO 55 G=1,NAPT-1
             DO 555 GG=1,NAPT
                IF ((CHECK/STAGES).EQ. GG) WAIT=W(G)+WAIT
555          CONTINUE
             CHECK=CHECK+1.
55        CONTINUE
*          PRINT*,'OVERTIME IS',W(N)
*          WRITE(2,*) 'OVERTIME IS', W(NAPT)
*           IF (X(A,NAPT).EQ. 0.) PLACE=NSLT
*           IF (X(A,NAPT).EQ. 0.) GOTO 930
            DO 929 SS=1, NSLT
              IF (X(A,NAPT).EQ.SS) PLACE= NSLT-X(A,NAPT)+1.
929        CONTINUE
*           PRINT*, 'PLACE=    ', PLACE
930          OT=(W(NAPT-1.)+PLACE+1./U)*C
*         PRINT*,'WAITING TIME IS', WAIT, NAPT,PLACE,U
*          PRINT*, 'OT COST IS', OT,A
           VAL=WAIT+OT
*           PRINT*, 'thebestisfirsttime', VAL
*           PRINT*, 'thebestis', VAL
           IF ((VAL.GE.BEST).OR. (VAL.EQ.0)) GOTO 2000
           BPLACE=PLACE
           BEST=VAL
           BWAIT=WAIT
           BOT=OT
*           PRINT*, 'thebestis', VAL
           DO 241 MM=1,NAPT
              X(N+1,MM)=X(A,MM)
241        CONTINUE
*           PRINT*, 'iswitched'
*          IF (VAL.LT.BEST) NUM=A
*          IF (VAL.LT.BEST) BEST=VAL
*          WRITE(2,*) 'WAITING TIME IS', WAIT
*         WRITE(2,*) 'OVERTIME COST IS', OT
*          WRITE(2,*) 'TOTAL COST FOR SCHEDULE', A, ' =    ', WAIT+OT
*          PRINT*,  'TOTAL COST FOR SCHEDULE', A, ' =    ', WAIT+OT
*           WRITE(2,*) ' '
```

```
*          WRITE(2,*) NUM,  BEST
*          PRINT*, 'iendedupat2000'
2000      CONTINUE
*          IF (BEST.GE.OLDBEST) PRINT*, 'hallelula'
          IF (BEST.GE.OLDBEST) GOTO 131
*          PRINT*, 'almost made it'
*          PRINT*, 'II', X(9,1),X(9,2),X(9,3),X(9,4),X(9,5)
*          PRINT*, X(9,6),X(9,7),X(9,8)
          DO 30 GG=1, N
              X(GG,1)=X(N+1.,1.)
*              PRINT*, 'X(GG,1.)=', X(GG,1.)
              DO 40 HH=2,NAPT-1
                 X(GG,HH)=X(N+1.,HH)
*                  PRINT*, 'X(GG,HH)=', X(GG,HH)
40            CONTINUE
              X(GG,NAPT)=X(N+1.,NAPT)
*              PRINT*, 'X(GG,NAPT)=', X(GG,NAPT)
30         CONTINUE
*
*          PRINT*, 'i gotta best solution'
*          WRITE(2,*) X(9,1),X(9,2),X(9,3),X(9,4),X(9,5)
*         WRITE(2,*) X(9,6),X(9,7),X(9,8)
*          PRINT*, X(9,1),X(9,2),X(9,3),X(9,4),X(9,5),X(9,6),X(9,7),X(9,8)
          DO 747 PP=1,N-1.
          IF (X(N+1,STAGES*PP) .EQ. 0.)  GOTO 747
              X(PP,PP*STAGES)=X(PP,PP*STAGES)-1.
              X(PP,PP*STAGES+STAGES)=X(PP,PP*STAGES+STAGES)+1.
*              PRINT*, X(PP,PP*STAGES),  X(PP,PP*STAGES+STAGES),PP
747       CONTINUE
*129       PRINT*,'OLDBEST=', OLDBEST, 'BEST=', BEST
*          PRINT*, (X(N+1,LL),LL=1,NAPT)
          WRITE(2,*) 'dabest:  ', BEST, 'place=  ', BPLACE
           PRINT*, 'dabest:', BEST, 'place=   ', BPLACE
            PRINT*, 'X(A,NAPT)', X(A,NAPT)
*          WRITE(2,*) (X(N+1,BB),BB=1,NAPT)
          OLDBEST=BEST
*          BWAIT=WAIT
*          BOT=OT
          GOTO  130
131       PRINT*, 'youreatheend'
          PRINT*, 'daverybestis', BEST
          WRITE(2,*) (X(N+1,RR),RR=1,NAPT)
          WRITE(2,*) BEST, 'theverybest', 'WAIT=', BWAIT, 'OT=', BOT
          PRINT*, (X(N+1,TT),TT=1,NAPT)
789       FORMAT(F5.1)
          STOP
          END
```

```
*** THIS PROGRAM APPROXIMATES THE EARLIEST CANDIDATE
*** SCHEDULE FOR THE CONTINUOS CASE
        REAL*8 X(200,200),Y(200,200),P(200,200)
        REAL*8 Q(200,200),W(200)
        REAL*8 U,SUM,SUM1,SUM2, GG
        REAL*8 WAIT,STORE,C,OT,TEMP,FACT,BEST,VAL,OLDBEST,TOTAL,N
        REAL*8 NAPT,NSLT,CHECK,CUSMU,STAGES,TRIAL,BWAIT,BOT
        INTEGER I,A,J,D,L,K,M,Z,COUNT
        OPEN (UNIT=2, FILE='realcont.out')
        BEST=100000.
        OLDBEST=100000.
        COUNT=1
        TRIAL=1.
        N=5.
        NSLT=7.
        C=4.
        STAGES=1.
        CUSMU=1.
        WRITE(2,*) STAGES, '-Erlang' ,' U=', CUSMU, N, '  CUSTOMERS'
        WRITE(2,*) 'Cost Factor =  ', C,'server cost is overtime'
        WRITE(2,*) '# of slots =   ', NSLT
        WRITE(2,*) ' '
        WRITE(2,*) '**********************************************'
        WRITE(2,*) ' '
        GOTO 787
767     NSLT=NSLT*2.
        TRIAL=TRIAL*2.
        COUNT=COUNT+1
        CUSMU=CUSMU/2.
        BEST=100000.*TRIAL
        OLDBEST=100000.*TRIAL
787     U=CUSMU*STAGES
        TOTAL=N+1
        NAPT=N*STAGES
        DO 1 F=1,NAPT+1.
            DO 2 G=1,F
             P(F,G)=0.
             Q(F,G)=0.
2            CONTINUE
1        CONTINUE
        IF(TRIAL.GT.1.) GOTO 420
        DO 3 G=1, TOTAL
            DO  434 CC=1, STAGES-1.
               X(G,CC)=0.
434         CONTINUE
            X(G,STAGES)=NSLT-1.
            DO 4 H=STAGES+1.,NAPT-1.
            X(G,H)=0.
4           CONTINUE
            X(G,NAPT)=1.
3       CONTINUE
        GOTO 130
420     DO 421 DD=1,TOTAL
            X(DD,1.)=X(DD,1.)*2.+2.
          DO 422 EE=2,N-1.
               X(DD,EE)=2.*X(DD,EE)
422       CONTINUE
            IF  (X(DD,N).EQ.1) X(DD,N)=X(DD,N)*2.
            IF  (X(DD,N).GT.1) X(DD,N)=X(DD,N)*2.-2.
421     CONTINUE
130     DO 2000 A=1,N-1.
            DO 11 B=1,NAPT
                Y(A,B)=X(A,B)*U
11          CONTINUE
*        PRINT*, 'i initialized everything'
        Z=0
```

```
*         next calculate the P probabilities
          DO 201 I=1,NAPT
              Z=Z+1
              SUM=0.
              FACT=1.
*
              DO 101 J=1,I
                 P(I,J)=EXP(-Y(A,Z))*((Y(A,Z)**(J-1))/FACT)
                 SUM=SUM+P(I,J)
                 FACT=FACT*(J)
101           CONTINUE
             P(I,I+1)=1.-SUM
201       CONTINUE
*
*         calculate the Q probabilities
          Q(1,2)=P(1,1)
          Q(1,1)=1-Q(1,2)
          DO 10 K=2,NAPT
            SUM1=0.
            DO 12 L=1,K
               SUM2=0.
               TEMP=0.
               D=1
               DO 14 M=L, K
                 TEMP=P(K,D)*Q(K-1,M)
                 SUM2=SUM2+TEMP
                 D=D+1
14             CONTINUE
            Q(K,L+1)=SUM2
            SUM1=SUM1+SUM2
12          CONTINUE
            Q(K,1)=1.-SUM1
10        CONTINUE
          DO 7 R=1,NAPT
            W(R)=0.
7         CONTINUE
          DO 33 E=1,NAPT
             STORE=0.
             DO 44 F=2,E+1
                STORE=Q(E,F)*(F-1)/U
                W(E)=(W(E)+STORE)
44           CONTINUE
33        CONTINUE
          WAIT=0.
          OT=0.
          CHECK=1.
          DO 55 G=1,NAPT-1
             DO 555 GG=1.,NAPT
                IF ((CHECK/STAGES).EQ. GG) WAIT=W(G)+WAIT
555          CONTINUE
             CHECK=CHECK+1.
55        CONTINUE
          OT=W(NAPT)*C
           VAL=WAIT+OT
           IF ((VAL.GE.BEST) .OR. (VAL.EQ.0)) GOTO 2000
           BEST=VAL
           BWAIT=WAIT
           BOT=OT
           DO 241 MM=1,NAPT
              X(N+1,MM)=X(A,MM)
241        CONTINUE
2000      CONTINUE
          IF (BEST.GE.OLDBEST) GOTO 131
          DO 30 GG=1, N
              X(GG,1.)=X(N+1.,1.)
              DO 40 HH=2,NAPT-1
```

```
                    X(GG,HH)=X(N+1.,HH)
40               CONTINUE
                 X(GG,NAPT)=X(N+1.,NAPT)
30            CONTINUE
*
             DO 747 PP=1,N-1.
             IF (X(N+1.,STAGES*PP) .EQ. 0.)  GOTO 747
                 X(PP,PP*STAGES)=X(PP,PP*STAGES)-1.
                 X(PP,PP*STAGES+STAGES)=X(PP,PP*STAGES+STAGES)+1.
747          CONTINUE
             PRINT*, 'bestis', BEST/TRIAL
             OLDBEST=BEST
             GOTO  130
131          PRINT*, 'youreatheend'
             PRINT*, 'verybestis', BEST/TRIAL
             PRINT*,  BWAIT/TRIAL, BOT/TRIAL
             PRINT*, (X(N+1,TT),TT=1,NAPT)
             PRINT*, 'count=    ', COUNT
             WRITE(2,*) (X(N+1,RR),RR=1,NAPT)
             WRITE(2,*) 'verybest=    ', BEST/TRIAL,'  COUNT=    ', COUNT
             WRITE(2,*) 'WAIT=',BWAIT/TRIAL,'OT=',BOT/TRIAL
             WRITE(2,*) ' '
             IF(1/NSLT.GT.0) GOTO 767
*             IF(NSLT.LT.1000000000) GOTO 767
789          FORMAT(F5.1)
             STOP
             END
```

```
*** THIS PROGRAM APPROXIMATES THE LATEST CANDIDATE SCHEDULE
*** FOR THE CONTINUOUS PROBLEM
        REAL*8 X(200,200),Y(200,200),P(200,200)
        REAL*8 Q(200,200),W(200)
        REAL*8 U,SUM,SUM1,SUM2, GG,TP
        REAL*8 WAIT,STORE,C,OT,TEMP,FACT,BEST,VAL,OLDBEST,TOTAL,N
        REAL*8 NAPT,NSLT,CHECK,CUSMU,STAGES,TRIAL,BWAIT,BOT
        INTEGER I,A,J,D,L,K,M,Z,COUNT
        OPEN (UNIT=2, FILE='1realcont.out')
        BEST=100000.
        OLDBEST=100000.
        COUNT=1
        TRIAL=1.
        N=5.
        NSLT=7.
        C=4.
        STAGES=1.
        CUSMU=1.
        WRITE(2,*) STAGES, '-Erlang' ,' U=', CUSMU, N, '  CUSTOMERS'
        WRITE(2,*) 'Cost Factor =  ', C,'server cost is overtime'
        WRITE(2,*) '# of slots =   ', NSLT
        WRITE(2,*) ' '
        WRITE(2,*) '************************************************'
        WRITE(2,*) ' '
        GOTO 787
767     NSLT=NSLT*2.
        TRIAL=TRIAL*2.
        COUNT=COUNT+1
        CUSMU=CUSMU/2.
        BEST=100000.*TRIAL
        OLDBEST=100000.*TRIAL
787     U=CUSMU*STAGES
        TOTAL=N+1
        NAPT=N*STAGES
        DO 1 F=1,NAPT+1.
            DO 2 G=1,F
             P(F,G)=0.
             Q(F,G)=0.
2            CONTINUE
1        CONTINUE
        IF(TRIAL.GT.1.) GOTO 445
        DO 3 G=1, TOTAL
            DO  434 CC=1, NAPT-1.
               X(G,CC)=0.
434         CONTINUE
            X(G,NAPT)=NSLT
3       CONTINUE
        GOTO 130
445       TP=0.
          TP=TP+1.
         IF(X(N+1.,TP).EQ. 0.) GOTO 445
420        DO 421 DD=1.,TOTAL
            X(DD,TP)=X(DD,TP)*2.- 2.
           DO 422 EE=TP+1.,N-1.
              X(DD,EE)= X(DD,EE) * 2.
422        CONTINUE
           X(DD,N)=2.*X(DD,N)+2.
421     CONTINUE
130     DO 2000 A=1,N-1.
            DO 11 B=1,NAPT
                Y(A,B)=X(A,B)*U
11          CONTINUE
        Z=0
*       next calculate the P probabilities
        DO 201 I=1,NAPT
            Z=Z+1
```

```
      SUM=0.
      FACT=1.
*
      DO 101 J=1,I
         P(I,J)=EXP(-Y(A,Z))*((Y(A,Z)**(J-1))/FACT)
         SUM=SUM+P(I,J)
         FACT=FACT*(J)
101   CONTINUE
      P(I,I+1)=1.-SUM
201   CONTINUE
*
*     calculate the Q probabilities
      Q(1,2)=P(1,1)
      Q(1,1)=1-Q(1,2)
      DO 10 K=2,NAPT
         SUM1=0.
         DO 12 L=1,K
            SUM2=0.
            TEMP=0.
            D=1
            DO 14 M=L, K
               TEMP=P(K,D)*Q(K-1,M)
               SUM2=SUM2+TEMP
               D=D+1
14          CONTINUE
         Q(K,L+1)=SUM2
         SUM1=SUM1+SUM2
12       CONTINUE
         Q(K,1)=1.-SUM1
10    CONTINUE
      DO 7 R=1,NAPT
        W(R)=0.
7     CONTINUE
      DO 33 E=1,NAPT
         STORE=0.
         DO 44 F=2,E+1
            STORE=Q(E,F)*(F-1)/U
            W(E)=(W(E)+STORE)
44       CONTINUE
33    CONTINUE
      WAIT=0.
      OT=0.
      CHECK=1.
      DO 55 G=1,NAPT-1
         DO 555 GG=1.,NAPT
            IF ((CHECK/STAGES).EQ. GG) WAIT=W(G)+WAIT
555      CONTINUE
         CHECK=CHECK+1.
55    CONTINUE
      OT=W(NAPT)*C
       VAL=WAIT+OT
       IF ((VAL.GE.BEST) .OR. (VAL.EQ.0)) GOTO 2000
       BEST=VAL
       BWAIT=WAIT
       BOT=OT
       DO 241 MM=1,NAPT
          X(N+1,MM)=X(A,MM)
241    CONTINUE
2000  CONTINUE
*      IF (BEST.GE.OLDBEST) PRINT*, 'hallelula'
      IF (BEST.GE.OLDBEST) GOTO 131
      DO 30 GG=1, N
          X(GG,1.)=X(N+1.,1.)
*
*
          DO 40 HH=2,NAPT-1
```

```
            X(GG,HH)=X(N+1.,HH)
40        CONTINUE
          X(GG,NAPT)=X(N+1.,NAPT)
30      CONTINUE
*
        DO 1747 PP=1,N-1.
          DO 888 JJ=1,N-1.
          IF (X(PP,PP*STAGES+JJ*STAGES) .EQ. 0.) GOTO 888
          X(PP,PP*STAGES+JJ*STAGES)=X(PP,PP*STAGES+JJ*STAGES)-1.
         X(PP,PP*STAGES)=X(PP,PP*STAGES)+1.
         GOTO 1747
888       CONTINUE
1747    CONTINUE
       PRINT*, 'bestis', BEST/TRIAL
*       WRITE(2,*)  'best',BEST/TRIAL
       OLDBEST=BEST
       GOTO  130
131    PRINT*, 'youreatheend'
       PRINT*, 'verybestis', BEST/TRIAL
       PRINT*,  BWAIT/TRIAL, BOT/TRIAL
       PRINT*, (X(N+1,TT),TT=1,NAPT)
       PRINT*, 'count=   ', COUNT
       WRITE(2,*) (X(N+1,RR),RR=1,NAPT)
       WRITE(2,*) 'verybest=    ', BEST/TRIAL,'  COUNT=    ', COUNT
       WRITE(2,*) 'WAIT=',BWAIT/TRIAL,'OT=',BOT/TRIAL
       WRITE(2,*) ' '
       IF(1/NSLT.GT.0) GOTO 767
*       IF(NSLT.LT.100) GOTO 767
789    FORMAT(F5.1)
       STOP
       END
```

Bibliography

1. Wang, P., "Static and Dynamic Scheduling of Customer Arrivals to a Single-Server System," Naval Logistics Research Quarterly, Vol 40, 1993, pp345-360.
2. Naor, P., "On the Regulation of Queue Size by Levying Tolls,"Econometrica, Vol.37, 1969, pp15-24.
3. Knudsen, N.C., "Individual and Social Optimization in a Multiserver Queue with a General Cost-Benefit Structure," Econometrica, Vol. 40, 1972, pp515-528.
4. Yechiali, U., "On Optimal Balking Rules and Toll Charges in a $GI/M/1$ Queueing Process," Operations Research, Vol. 19, 1979, pp349-370.
5. Yechiali, U., "Customers Optimal Joining Rules for the $GI/M/s$ Queue," Management Science, Vol. 18, 1972, pp434-443.
6. Rue, R.C., and M. Rosenshine, "Optimal Control for Entry of Many Classes of Customers to an M/M/1 Queue," Naval Logistics Research Quarterly, Vol. 28, 1981, pp489-495.
7. Rue, R.C., and M. Rosenshine, "Optimal Control of Entry Classes to an $M/E_k/1$ Queue Serving Several Classes of Customers," Naval Logistics Research Quarterly, Vol. 30, 1983, pp217-226.
8. Healy, K. J., C. D. Pegden, and M. Rosenshine, "Scheduling Arrivals to Multiple Server Queues," Working Paper No. 82-128, Department of Industrial and Management Systems Engineering, The Pennsylvania State University, October 1982.
9. Liao, C.- J.,"Planning Timely Arrivals to Stochastic Production or Service Systems," PhD Thesis, Department of Industrial and Management Systems Engineering, The Pennsylvania State University, August 1988.
10. Ross, S., A First Course in Probability Theory, MacMillan, New York, New York, 1988, pp13-16.

Vita

John Simeoni was born in Chicago, Illinois in 1959. He received a Bachelor of Music degree from Northwestern University in 1981. After several years as a professional musician and educator, he returned to school and received a Master of Mathematics degree from California State University Sacramento in 1988. He joined the Air Force in 1989, and worked as a Contract Business Manager for the Consolidated Space Operations Center Program Office at Los Angeles Air Force Base (LAAFB), California prior to coming to AFIT. While at LAAFB, he received a Master of Business Administration degree from Chapman University (1992).

Permanent address: P.O. Box 33845
Dayton, Ohio 45433

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1994	Master's Thesis

4. TITLE AND SUBTITLE
An Efficient Approach to Solving the Optimal Control of Arrivals Problem

5. FUNDING NUMBERS

6. AUTHOR(S)
John R. Simeoni, Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
WPAFB, OH 45433-6503

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GOR/ENS/94-14

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The optimal control of arrivals problem is one which has many applications in both defense and industry. Simply stated, the problem addresses how to schedule a finite number of customers in a finite number of equal-length time slots, where each customer's service time comes from a specified probability distribution. There are two cost components, one based on total expected customer waiting time and the other based on the expected amount of time the server stays open beyond its scheduled completion time. Currently, solutions have been developed to the optimal control of arrivals problem, but they are computationally slow and only work for exponential distributions. This thesis presents an algorithm for the optimal control of arrivals problem which is both computationally efficient and works for r-Erlang distributions.

14. SUBJECT TERMS
Optimal control of arrivals, planning arrivals, scheduling theory, queueing theory, control of queues

15. NUMBER OF PAGES
54

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL